Heuristiken zur multikriteriellen Komposition von Diensten in dienstbasierten Informationssystemen

René Ramacher
Lars Mönch

Veröffentlicht in:
Multikonferenz Wirtschaftsinformatik 2012
Tagungsband der MKWI 2012
Hrsg.: Dirk Christian Mattfeld; Susanne Robra-Bissantz

Braunschweig: Institut für Wirtschaftsinformatik, 2012

Heuristiken zur multikriteriellen Komposition von Diensten in dienstbasierten Informationssystemen

René Ramacher

FernUniversität in Hagen, Lehrstuhl für Unternehmensweite Softwaresysteme,
58084 Hagen, E-Mail: rene.ramacher@fernuni-hagen.de

Lars Mönch

FernUniversität in Hagen, Lehrstuhl für Unternehmensweite Softwaresysteme,
58084 Hagen, E-Mail: lars.moench@fernuni-hagen.de

Abstract

Service-orientierte Architekturen unterstützen die Bereitstellung von Anwendungsfunktionalität durch Dienstkomposition. Dabei werden nicht-funktionale Attribute betrachtet, um zwischen funktional gleichwertigen Diensten zu unterscheiden. Wir untersuchen die Auswahl von Diensten aus einer Menge von Dienstkandidaten für den Fall einer sequentiellen Komposition, so dass die Kosten des komponierten Dienstes eine vorgegebene Schranke nicht überschreiten und gleichzeitig die Ausführungszeit minimiert und die Verfügbarkeit des komponierten Dienstes maximiert wird. Da dieses Problem NP-schwer ist, wird ein genetischer Algorithmus zur Ermittlung der Menge von Pareto-optimalen Lösungen vorgeschlagen, der mit problemspezifischen Heuristiken kombiniert wird. Die Ergebnisse numerischer Experimente mit zufällig erzeugten Probleminstanzen zeigen die Leistungsfähigkeit des Ansatzes.

1 Einleitung und Motivation

Service-orientierte Architekturen (SOA) werden in jüngerer Zeit als Architekturparadigma für unternehmensweite IT-Systeme vorgeschlagen, um deren Ausrichtung an den Geschäftsprozessen eines Unternehmens zu verbessern [10]. Das SOA-Paradigma sieht die Bereitstellung von IT-Leistungen in Form von lose gekoppelten Diensten vor. Neue Dienste können durch die Komposition von existierenden Diensten realisiert werden. Durch die Komposition von Diensten erfolgt eine Wiederverwendung von existierender Funktionalität auf hohem Abstraktionsniveau, wodurch die Anpassung der bereitgestellten IT-Funktionen im Idealfall beschleunigt und deren Qualität gesteigert werden kann [9,10].

Die lose Kopplung von Diensten ermöglicht eine dynamische Bindung von Diensten, wobei eine solche Bindung bedeutet, dass die zur Realisierung einer Komposition erforderlichen Dienste nicht bereits zur Entwurfszeit fest an die Dienstkomposition gebunden werden, sondern dass zu

diesem Zeitpunkt lediglich eine abstrakte Beschreibung der im Rahmen der Komposition durchzuführenden Aufgaben und der für die Aufgabenlösung erforderlichen Funktionalität erfolgt. Die Bindung an konkrete Dienste zur Erfüllung der Aufgaben erfolgt zur Ausführungszeit des komponierten Dienstes. Viele Fragestellungen der dynamischen Dienstkomposition sind bisher nicht ausreichend geklärt [9].

Bei der Beschreibung von Diensten werden auch nicht-funktionale Attribute zur Beschreibung der Dienstgüte, als Quality-of-Service (QoS) bezeichnet, und zur Abbildung von Zusicherungen an die Ausführungsbedingungen eines Dienstes (Service-Level-Agreement) verwendet. QoS-Attribute dienen der Differenzierung zwischen funktional gleichwertigen Diensten. Wichtige QoS-Attribute eines Dienstes sind seine Kosten sowie die Ausführungsgeschwindigkeit und die Verfügbarkeit des Dienstes. Die Dienstgüte einer Komposition wird von den QoS-Eigenschaften der Dienste bestimmt, die zur Realisierung der Dienstkomposition verwendet werden. Für die Kosten einer Dienstkomposition sind häufig Schranken vorgegeben, die nicht überschritten werden dürfen, während Diensteigenschaften wie die Ausführungsgeschwindigkeit oder die Verfügbarkeit zu optimieren sind.

Optimierungsprobleme mit mehrfachen Zielsetzungen können unter Verwendung einer gewichteten Zielfunktion gelöst werden, wenn Informationen bezüglich der Wichtigkeit der Zielsetzungen bekannt sind, d. h., wenn Präferenzinformationen a priori vorliegen [3]. Solche Informationen liegen aber im Falle einer dynamischen Dienstkomposition typischerweise nicht vor. Das Ziel dieser Arbeit besteht folglich darin, Verfahren bereitzustellen, die eine Optimierung von mehrfachen Zielsetzungen ermöglichen, ohne die Ziele in eine Zielfunktion zu integrieren. Da typischerweise bei der Betrachtung von konkurrierenden Zielsetzungen keine eindeutige Lösung existiert, zielt ein solches Verfahren darauf ab, eine Menge von alternativen Lösungen zu bestimmen. Aus der Menge dieser Lösungen kann dann durch den Entscheider bzw. die Entscheidungseinheit im automatisierten Fall, in [4] werden dazu Softwareagenten vorgeschlagen, eine geeignete Dienstkomposition ausgewählt werden. In der Literatur sind bisher derartige Ansätze für Dienstkompositionsfragestellungen nur unzureichend betrachtet worden (vergl. [12] für einen solchen Ansatz). Insbesondere fehlt häufig eine Leistungsbewertung. In dieser Arbeit werden Heuristiken vorgeschlagen, die den generischen Non-Dominated–Sorting-Genetic-Algorithm II (NSGA-II) [1] mit problemspezifischen Verfahren kombinieren. Eine umfassende Leistungsbewertung auf Basis zufällig erzeugter Probleminstanzen erfolgt, die zeigt, dass die vorgeschlagenen Verfahren Dienstkompositionen hoher Güte mit akzeptablem Zeitaufwand finden.

Die Arbeit ist wie folgt aufgebaut. Das untersuchte Dienstkompositionsmodell wird in Kapitel 2 beschrieben. Das zu lösende Problem wird als binäres Optimierungsproblem formuliert. Vorarbeiten werden diskutiert. Heuristiken zur Lösung des Problems werden in Kapitel 3 vorgeschlagen. Die Leistungsfähigkeit dieser Heuristiken wird in Kapitel 4 untersucht.

2 Dienstkompositionsmodell und Problemstellung

Wir betrachten in dieser Arbeit sequentielle Dienstkompositionen, bei denen n Aufgaben, als Tasks t_i bezeichnet, nacheinander ausgeführt werden. Ein Task ist eine Abstraktion von einem konkreten Dienst mit bestimmten funktionalen Anforderungen. Jedem Task $t_i, i = 1, \ldots, n$ ist eine Menge von Diensten S_i zugeordnet, welche die funktionalen Anforderungen von t_i erfüllen.

Um eine Dienstkomposition auszuführen, ist es erforderlich, an jeden Task t_i einen Dienst aus S_i zu binden. Ein Dienst, der an einem Task t_i gebunden ist, wird mit s_i bezeichnet. In bestimmten Situationen ist es erforderlich, zwischen den unterschiedlichen Diensten aus S_i zu unterscheiden. Dazu bezeichnen wir diese mit $s_{ij}, j = 1,\ldots,|S_i|$. Eine Dienstkomposition, bei der an jeden Task genau ein Dienst gebunden ist, wird als konkrete Dienstkomposition bezeichnet. Wir bezeichnen die Menge aller Dienste und Dienstkompositionen mit S. Eine konkrete Dienstkomposition s hat somit in dieser Arbeit die Form $s =: (s_1,\ldots,s_n)$ mit $s_i \in S_i$.

Zur Bewertung von Diensten und Dienstkompositionen werden QoS-Attribute betrachtet. Dabei wird Attribut k durch eine Bewertungsfunktion q_k repräsentiert, die jedem Dienst aus $S_i, i = 1,\ldots,n$ eine reelle Zahl zuordnet. Wir nehmen an, dass die Funktionen q_k bekannt sind. Der Wert eines QoS-Attributs $q_k(s)$ einer Dienstkomposition s wird in dieser Arbeit aus den QoS-Attributen der ausgewählten, d.h. an die n Tasks gebundenen Dienste, durch Aufsummieren der Attributwerte der Dienste berechnet. Wir erhalten somit:

$$q_k(s) := \sum_{i=1}^{n} q_k(s_i). \qquad (1)$$

Wir betrachten in dieser Arbeit ein Modellproblem, das die QoS-Attribute Ausführungszeit und Verfügbarkeit des jeweiligen Dienstes sowie die Kosten, die bei der Ausführung eines Dienstes entstehen, umfasst. Es ist leicht einzusehen, dass für einen komponierten Dienst die Attribute Ausführungszeit und Kosten entsprechend Gleichung (1) aus den entsprechenden Attributwerten der n Dienste berechnet werden können. Im Falle der Verfügbarkeit ergibt sich die Verfügbarkeit der Dienstkomposition als Produkt der Verfügbarkeiten der n Dienste, die zur Realisierung der Dienstkomposition verwendet werden. Durch Logarithmieren dieses Produktes kann aber auch in diesem Fall eine Bewertung wie in Gleichung (1) erreicht werden. Dem Attribut Ausführungszeit wird die Bewertungsfunktion q_1 zugeordnet, dem Attribut Verfügbarkeit die Funktion q_2 sowie dem Attribut Kosten die Funktion q_3.

Die Auswahl der an die n Tasks zu bindenden Dienste soll so erfolgen, dass die Attribute Kosten und Verfügbarkeit minimiert bzw. maximiert werden sollen, wobei die Maximierung durch Betrachten der jeweiligen negativen Bewertungsfunktion auf eine Minimierung zurückgeführt werden kann. Für das Kostenattribut fordern wir, dass für alle zulässigen Dienstkompositionen ein vorgegebener Wert Q_{max} nicht überschritten werden darf.

Prinzipiell können Optimierungsaufgaben mit mehrfacher Zielsetzung durch Zielgewichtung behandelt werden [3]. Falls t Ziele f_i zu berücksichtigen sind, betrachtet man die gewichtete Zielfunktion $f := \sum_{i=1}^{t} w_i f_i$ mit den Gewichten $w_i \geq 0$ und $\sum_{i=1}^{t} w_i = 1$. Die Gewichte müssen dabei a priori bekannt sein. Das betrachtete Dienstkompositionsmodell kann unter Verwendung einer solchen gewichteten Zielfunktion wie folgt als binäres lineares Optimierungsproblem formuliert werden. Die folgenden Indizes und Mengen werden im Modell verwendet:

$i = 1,\ldots,n$: Index für Tasks,

S_i: Menge der Dienstkandidaten für Task t_i,

$j = 1,\ldots,|S_i|$: Index für Dienste, die zur Ausführung von Task t_i zur Verfügung stehen,

$k = 1,2,3$: Index für QoS-Attribute.

Das Modell verwendet die folgenden Parameter:

$q_k(s_{ij})$: Wert des QoS-Attributs k für Dienst $s_{ij} \in S_i$,

Q_{max}: maximal möglicher Attributwert für das Kostenattribut einer Komposition,

$w_k, k = 1,2$: Gewicht für Attribut k in der gewichteten Zielfunktion.

Die folgenden Entscheidungsvariablen werden verwendet:

$$z_{ij} := \begin{cases} 1, \text{falls Dienst } s_{ij} \text{ für Task } t_i \text{ ausgewählt wird} \\ 0, \text{sonst} \end{cases} . \quad (2)$$

Das Modell kann nun wie folgt formuliert werden:

Minimiere

$$\sum_{i=1}^{n} \sum_{j=1}^{|S_i|} \left(w_1 q_1(s_{ij}) + w_2 q_2(s_{ij}) \right) z_{ij} \quad (3)$$

unter den Nebenbedingungen:

$$\sum_{j=1}^{|S_i|} z_{ij} = 1, \quad i = 1,\ldots,n, \quad (4)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{|S_i|} q_3(s_{ij}) z_{ij} \leq Q_{max}, \quad (5)$$

$$z_{ij} \in \{0,1\}, \quad i = 1,\ldots,n, j = 1,\ldots,|S_i| . \quad (6)$$

Die Zielfunktion (3) versucht, die gewichtete Summe aus Ausführungszeit und negativer logarithmierter Verfügbarkeit zu minimieren. Nebenbedingung (4) stellt sicher, dass für jeden Task t_i genau ein Dienst aus der Menge S_i ausgewählt wird. Durch Nebenbedingung (5) wird abgebildet, dass die Kosten für die Durchführung der resultierenden Dienstkomposition beschränkt sind. Nebenbedingung (6) stellt schließlich sicher, dass die Entscheidungsvariablen nur binäre Werte annehmen. Wir bemerken, dass ein Multiple-Choice-Knapsack-Problem (MCKP) vorliegt. Dieses Problem ist NP-schwer [6]. Demzufolge ist auch das von uns untersuchte Dienstkompositionsproblem NP-schwer.

Für Zielfunktion (3) ist erforderlich, dass die Gewichte w_i vorab bekannt sind. Wir können auf die gewichtete Zielfunktion (3) verzichten, wenn wir alle Pareto-optimalen Dienstkompositionen suchen, die Nebenbedingung (4) und (5) erfüllen. Allgemein ist eine zulässige Lösung $x \in X$ einer Optimierungsaufgabe mit t Zielsetzungen Pareto-optimal, wenn in der Menge aller zulässigen Lösungen X keine Lösung x' existiert, so dass $f_i(x') \leq f_i(x)$ für alle $i = 1,\ldots,t$ gilt und

in mindestens einer dieser Ungleichungen keine Gleichheit gilt [3]. Die Menge aller Pareto-optimalen Lösungen wird als Pareto-Front bezeichnet.

In dieser Arbeit schlagen wir Heuristiken zur Ermittlung der Pareto-Front für das Dienstkompositionsproblem (Service Composition Problem – SCP) vor, wobei die Nebenbedingungen (4) und (5) erfüllt sein müssen. Wir bemerken, dass das von uns betrachtete Problem leicht verallgemeinert werden kann, indem wir in Zielfunktion (3) anstelle der Bewertungsfunktionen für zwei QoS-Attribute $m \geq 2$ derartige Attribute zulassen. Außerdem ist es möglich, anstelle der einzelnen Nebenbedingung (5) für maximal zulässige Werte der QoS-Attribute allgemeiner $r \geq 1$ solcher Bedingungen zu verwenden. Wir verwenden die Bezeichnung $SCP(m,r,W)$ und $SCP(m,r,PF)$ für die jeweilige verallgemeinerte Problemstellung mit gewichteter Zielfunktion oder mit Pareto-Front.

Zen et al. [15] untersuchen eine Problemstellung vom Typ $SCP(m,r,W)$, bei der Dienstkompositionen mit Verzweigungen und Parallelisierungen betrachtet werden. Methoden der ganzzahligen Optimierung werden angewandt, um Lösungen zu ermitteln. Yu et al. [13,14] betrachten ebenfalls eine $SCP(m,r,W)$-Problemstellung. Exakte Verfahren werden untersucht, die im Wesentlichen darauf basieren, dass das Problem ähnlich zu einem MCKP ist. Aufgrund der NP-Schwere des Dienstkompositionsproblems werden Metaheuristiken in der Literatur untersucht. Ein genetischer Algorithmus zur Lösung von $SCP(m,1,W)$ wird in [5] verwendet, während für dieses Problems in [7] ein Ameisenalgorithmus vorgeschlagen wird.

Während die bisher diskutierten Arbeiten stets eine gewichtete Zielfunktion verwenden, wird in [8] eine $SCP(m,0,PF)$-Problemstellung behandelt. Eine NSGA-II-artige Heuristik wird vorgeschlagen, allerdings fehlen sowohl die Angabe der verwendeten Lösungsrepräsentation als auch Aussagen zur Leistungsfähigkeit der Heuristik. Zwei genetische Algorithmen für $SCP(m,r,PF)$ werden in [12] beschrieben. Das E^3-Verfahren verwendet eine spezielle Fitness-Funktion, um die Konvergenz des genetischen Algorithmus in Richtung Pareto-optimaler Lösungen zu fördern. Eine gleichmäßige Verteilung der gefundenen Lösungen auf der Pareto-Front wird angestrebt. Im Gegensatz dazu zielt das $X-E^3$-Verfahren auf die Ermittlung der Extremlösungen der Pareto-Front ab. Beide Verfahren, auch kombiniert, sind leistungsfähiger als NSGA-II. Lediglich Heuristiken werden miteinander verglichen. Eine Leistungsbewertung findet nur auf Basis einiger weniger Probleminstanzen statt.

In der vorliegenden Arbeit entwickeln wir neben Heuristiken auch exakte Verfahren, um zumindest für Probleminstanzen geringer Größe die Leistungsfähigkeit der vorgeschlagenen Heuristiken untersuchen zu können. Außerdem kombinieren wir problemspezifische Heuristiken mit dem NSGA-II-Verfahren und führen eine umfassende Leistungsbewertung auf Basis zufällig erzeugter Probleminstanzen durch.

3 Heuristischer Lösungsansatz

In diesem Kapitel wird ein Lösungsverfahren für $SCP(2,1,PF)$ beschrieben. Im Abschnitt 3.1 wird zunächst die NSGA-II-Methaheuristik und deren Anwendung zur Lösung von $SCP(2,1,PF)$ dargestellt. Anschließend werden problemspezifische Heuristiken zur Erweiterung von NSGA-II vorgeschlagen. Im Abschnitt 3.3 wird schließlich ein Verfahren, das Pareto-optimale Lösungen ermittelt, entwickelt.

3.1 NSGA-II-basierter Lösungsansatz

NSGA-II ist ein auf den Prinzipen von genetischen Algorithmen basierendes Verfahren zur Approximation einer Pareto-Front. Der elitäre Selektionsoperator des NSGA-II teilt die Lösungen einer Population zunächst in disjunkte Mengen ein, die als Fronten bezeichnet werden. Eine Front, der eine Lösung *y* zugeordnet wird, wird durch die Lösungen festgelegt, die *y* dominieren. Der Fitnesswert einer Lösung wird anhand der Front bestimmt, der eine Lösung zugeordnet ist. Um eine Streuung der Lösungen zu erreichen, wird ein Crowding-Operator verwendet, durch den Lösungen zusätzlich anhand ihres Abstands zu anderen Lösungen bewertet werden.

Um NSGA-II zur heuristischen Lösung von $SCP(2,1,PF)$ zu verwenden, muss eine geeignete Lösungsrepräsentation innerhalb der Chromosomen erfolgen. Wir verwenden dafür in dieser Arbeit ein Feld der Form $(j_1,\ldots,j_n)$. Der *i*-te Eintrag stellt dabei den Index des Dienstes $s_i \in S_i$ dar, der an Task t_i gebunden ist, wobei j_i eine natürliche Zahl mit $1 \leq j_i \leq |S_i|$ ist.

Ein Uniform-Crossover-Operator wird verwendet, der für jeden Feldeintrag des Kindchromosoms zunächst mit Wahrscheinlichkeit 0.5 eines der beiden Elternchromosomen zufällig auswählt und dann dessen Feldeintrag übernimmt. In den numerischen Experimenten war dieser Crossover-Operator anderen Operatoren wie zum Beispiel One-Point- oder Two-Point-Crossover überlegen. Als Mutationsoperator wird ein zufälliger Austausch zwischen Feldeinträgen verwendet. Dazu wird ein Chromosom mit einer Wahrscheinlichkeit p_m für eine Mutation ausgewählt. Für das *i*-te Feld wird mit einer Wahrscheinlichkeit von p_g zufällig ein neuer Dienst $s_i \in S_i$ ausgewählt, wobei die Dienste aus S_i gleichwahrscheinlich sind.

3.2 Problemspezifische Erweiterungen von NSGA-II

Aus der Literatur ist bekannt, dass die Leistungsfähigkeit von NSGA-II erhöht werden kann, wenn NSGA-II mit lokalen Suchverfahren kombiniert wird [2]. Deshalb schlagen wir die Anwendung von problemspezifischen Heuristiken innerhalb von NSGA-II für $SCP(2,1,PF)$ vor. Wir bezeichnen diese Heuristiken mit HR bzw. HI.

Durch eine zufällige Initialisierung der Startpopulation oder durch Anwendung der genetischen Operatoren können unzulässige Lösungen s entstehen, bei denen Nebenbedingung (5) verletzt ist. Die HR-Heuristik versucht, eine unzulässige Lösung $s = (s_1,\ldots,s_n)$ in eine zulässige zu überführen. Die Menge $\tilde{S} = \{s_{ij} | s_{ij} \in S_i,\ q_3(s_{ij}) < q_3(s_i)\}$ wird dazu betrachtet. Wir versuchen nun, Dienste aus $\tilde{S}$ so auszuwählen, dass der Wert für q_3 sich verringert, um Nebenbedingung (5) zu erfüllen, und gleichzeitig der Wert der zu minimierenden QoS-Attribute möglichst gering erhöht wird. Wir betrachten für jeden Dienst $s_{ij} \in \tilde{S}$ den folgenden Quotienten:

$$\mu(i,j) := \{w_1(q_1(s_{ij}) - q_1(s_i)) + w_2(q_2(s_{ij}) - q_2(s_i))\} / (q_3(s_i) - q_3(s_{ij})), \qquad (7)$$

wobei die w_i Gewichte sind. Wir wählen den Dienst $s_{ij} \in \tilde{S}$ aus, der zum kleinsten Wert von $\mu(i,j)$ führt. Falls mehrere solcher Dienste existieren, wird zufällig einer gewählt. Durch dieses Vorgehen wird sichergestellt, dass einerseits der Zielfunktionszuwachs gering ist, gleichzeitig aber q_3 möglichst stark verringert wird. Wenn die so gefundene Lösung s^* Nebenbedingung (5) erfüllt, wird das Verfahren beendet, ansonsten für s^* wiederholt. Da in $\tilde{S}$ alle möglichen Dienste

berücksichtigt werden, die den Wert von q_3 verringern, ist bei der Anwendung von HR garantiert, dass eine zulässige Lösung gefunden wird, falls eine solche existiert.

Für eine zulässige Lösung *s* existiert ein $\Delta \geq 0$, so dass $q_3(s) + \Delta = Q_{max}$ gilt. Die HI-Heuristik versucht, ein geeignetes Δ zu nutzen, um die Werte von q_1 und q_2 weiter zu verringern, ohne dass Δ negativ wird. Wir betrachten $\bar{S} = \{s_{ij} \mid q_k(s_{ij}) < q_k(s_i), k = 1,2\}$. Der Dienst $s_{ij} \in \bar{S}$ mit dem größten Wert für $\mu(i,j)$ wird ausgewählt. Anschließend wird s_{ij} an t_i gebunden und für die daraus resultierende Lösung s^* überprüft, ob $q_3(s^*) \leq Q_{max}$ gilt. Falls die Bedingung erfüllt ist, wird das Verfahren für s^* wiederholt, andernfalls wird der letzte Austausch rückgängig gemacht und HI beendet.

Zur Berechnung von $\mu(i,j)$ sind Gewichte w_i erforderlich. Diese werden für jedes s individuell abhängig von dessen Lage im Kriterienraum durch:

$$w_k(s) := \left(q_k^{max} - q_k(s)\right) / \left(q_k^{max} - q_k^{min}\right), \quad k = 1,2 \tag{8}$$

ermittelt, wobei mit q_k^{max} bzw. q_k^{min} der maximale bzw. minimal Wert des *k*-ten QoS-Attributs in der aktuellen Population bezeichnet werden. Gemäß der Wahl (8) werden für eine Lösung s besonders die Ziele stark priorisiert, deren Werte nah am individuellen Minimum des jeweiligen Ziels liegen. Diese Art der Gewichtswahl begünstigt eine vollständige Abtastung der Pareto-Front.

NSGA-II kann durch Aufnahme von Lösungen hoher Qualität in die Startpopulation beschleunigt werden. Dazu berechnen wir für eine lineare Kombination der Ziele optimale Lösungen, wobei q_3 zunächst ebenfalls als zu minimierendes QoS-Attribut aufgefasst wird. Die Zielfunktion für die lineare Kombination der Ziele lautet in diesem Fall:

$$\sum_{i=1}^{n} \left(w_1 q_1(s_i) + w_2 q_2(s_i) + w_3 q_3(s_i)\right) \tag{9}$$

Für einen gegebenen Gewichtsvektor $w := (w_1, w_2, w_3)$ lässt sich eine optimale Lösung leicht bestimmen, indem für jedes t_i der Dienst s_i gesucht wird, so dass der *i*-te Summand in (9) minimal wird. Durch $W_g := \{(k_1/g, k_1/g, k_1/g) \mid k_i \in \{0,\ldots,g\}, k_1 + k_2 + k_3 = g\}$ werden Gewichte gewählt, wobei durch die natürliche Zahl g die Anzahl der Gewichtsvektoren bestimmt wird. Sei s eine optimale Lösung für das Problem mit Zielfunktion (9) und $w \in W_g$. Auf s wird HR angewandt, um sicherzustellen, dass die Lösung zulässig ist. Außerdem werden Lösungen durch HI gegebenenfalls verbessert. Aus der Menge der so ermittelten Lösungen werden dominierte Lösungen entfernt. Die verbleibenden Lösungen werden in die Startpopulation aufgenommen. Falls die gewählte Populationsgröße so nicht erreicht wird, werden zufällig erzeugte Lösungen in die Population aufgenommen. Die Abkürzung NSGA-II-H wird für die Erweiterung von NSGA-II durch HR und HI verwendet.

3.3 Ermittlung von Pareto-optimalen Lösungen

Die in Abschnitt 3.1 und 3.2 vorgestellten Heuristiken bestimmen näherungsweise eine Pareto-Front, für deren Elemente dabei nicht gewährleistet ist, dass sie Pareto-optimal sind. Wir beschreiben nun ein Verfahren, das ausgehend von einer Approximation y_{approx} einer Pareto-Front

eine Front y_{opt} mit Pareto-optimalen Lösungen findet. Hierzu wird auf jede Lösung $s \in y_{approx}$ das folgende binäre Optimierungsmodell angewendet, das ausgehend von s eine Pareto-optimale Lösung s_{opt} mit $q_k(s_{opt}) \leq q_k(s)$ für $k = 1,2$ bestimmt. In Modell (3) – (6) wird Zielfunktion (3) durch $\sum_{i=1}^{n}\sum_{j=1}^{|S_i|} q_{m_l}(s_{ij})\, z_{ij}$ ersetzt. Außerdem wird das Modell durch die Nebenbedingungen (10) und (11) erweitert. Das resultierende Modell wird in zwei Iterationen $l = 1,2$ gelöst. Die Indizes $m_l \in \{1,2\}$ bestimmen das in der *l*-ten Iteration zu minimierenden QoS-Attribut. Mit s^{l-1} wird die Lösung der *l*-ten Iteration bezeichnet. Für die erste Iteration wird $s^0 := s$ für ein bestimmtes $s \in y_{approx}$ verwendet. Die beiden zusätzlichen Nebenbedingungen lauten:

$$\sum_{i=1}^{n}\sum_{j=1}^{|S_i|} q_1(s_{ij})\, z_{ij} \leq q_1(s^{l-1}), \tag{10}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{|S_i|} q_2(s_{ij})\, z_{ij} \leq q_2(s^{l-1}). \tag{11}$$

Die modifizierte Zielfunktion dient der Minimierung des Wertes von q_{m_l}. Durch die Bedingungen (10), (11) wird sichergestellt, dass der Wert der entsprechenden QoS-Attribute nicht größer ist als in der vorhergehenden Iteration. Es ist unmittelbar einzusehen, dass die Auswahl der m_l eine Priorisierung der QoS-Attribute zur Folge hat, da erwartet wird, dass für die Minimierung des Wertes des in der ersten Iteration gewählten QoS-Attributes aufgrund von $q_k(s^0) \geq q_k(s^1)$ mehr Optimierungsspielraum besteht. Zur Auswahl von m_l wird für eine Lösung s der Quotient (8) betrachtet. Mit der gleichen Argumentation wie in Abschnitt 3.2 wird das QoS-Attribut q_k mit dem größeren Wert für $w_k(s)$ höher priorisiert. Falls $w_k(s) = w_2(s)$ gilt, wird eines der beiden Attribute zufällig ausgewählt. Die Front y_{opt} kann weniger Lösungen enthalten als y_{approx}, da Lösungen aus y_{approx} zusammenfallen können. Wenn das Modell auf die durch NSGA-II-H gewonnene Pareto-Front angewendet wird, bezeichnen wir die Heuristik als IP.

4 Numerische Experimente

In Abschnitt 4.1 beschreiben wir die verwendeten Probleminstanzen und Leistungsmaße. Anschließend vergleichen wir die heuristischen Lösungen mit einer wahren Pareto-Front. Die Heuristiken werden untereinander in Abschnitt 4.3 verglichen.

4.1 Erzeugung von Probleminstanzen und Leistungsbewertung

Die Anzahl der Tasks und Dienstkandidaten werden als Faktoren für die Erzeugung von Probleminstanzen verwendet. Außerdem werden der Grad der Konvexität der zu erwarteten Pareto-Front für eine Probleminstanz sowie eine Konstante zur Festlegung von Q_{max} als zusätzliche Faktoren betrachtet. Tabelle 1 fasst das verwendete Versuchsdesign zusammen.

Zur Festlegung der Werte der QoS-Attribute wird für jeden Task zunächst die folgende Menge $V = \{(x_1, x_2, x_3) \mid x_i \in [10i, 100i], 486x_1 + 243x_2 + 162x_3 = 102060, x_1, x_2 \in \mathrm{IN}\}$ eingeführt. Die Punkte $(10,200,300)$, $(100,20,300)$, $(100,200,30)$ liegen in der V definierenden Ebene.

Die Funktion $t(x_1,x_2,x_3) = x_3 + r\,c\,\{(100-x_1)/270\}^2\,\{(200-x_2)/540\}^2\,\{(300-x_3)/810\}^2$ wird eingeführt, wobei $r \sim U(0,1)$ gilt und c die Krümmung der Fläche beeinflusst. Wir betrachten nun die Menge $V' := \{(x_1,x_2,t(x_1,x_2,x_3)) \mid (x_1,x_2,x_3) \in V\}$. Für jeden Dienst s_{ij} wählen wir dann zufällig die Attributwerte durch $(q_1(s_{ij}),q_2(s_{ij}),q_3(s_{ij})) \in V'$. Der Wert von Q_{max} in Nebenbedingung (5) ergibt sich für jede Probleminstanz in Abhängigkeit von *C* durch die Festlegung $Q_{max} := \sum_{i=1}^{n}\left(\min_{j\in\{1,\ldots,|S_i|\}} q_3(s_{ij}) + C\left(\max_{j\in\{1,\ldots,|S_i|\}} q_3(s_{ij}) - \min_{j\in\{1,\ldots,|S_i|\}} q_3(s_{ij}) \right)\right)$.

Faktor	Ausprägung	Anzahl
Anzahl der Tasks, *n*	10,20,30	3
Anzahl der Dienste pro Task, $\lvert S_i \rvert$	10,20,30	3
Grad der Konvexität, *c*	-4,-2,2,4	4
Faktor zur Festlegung von Q_{max}, *C*	2	1

Tabelle 1: Faktoren zur Generierung der Probleminstanzen

Um eine Lösungsmenge y_{known} und eine Referenzmenge y_{true} zu vergleichen, werden kardinalitäts- und distanzbasierten Leistungsmaße in Anlehnung an [11,16] verwendet. ONVG ist als die Anzahl der Elemente einer Pareto-Front definiert, d.h., es gilt $ONVG(y_{known}) := |y_{known}|$. Das $Error$-Maß bestimmt den relativen Fehler einer Pareto-Front im Vergleich zu y_{true} und ist durch $Error(y_{known}) := \sum_{i=1}^{|y_{known}|} e_i \Big/ |y_{known}|$ gegeben, wobei die Indikatorvariable e_i den Wert 1 annimmt, falls $y_i \notin y_{known} \cap y_{true}$ gilt. Außerdem betrachten wir die folgenden distanzbasierten Maße zur Bewertung einer Lösungsmenge im Vergleich zu einer Referenzmenge:

$$dist_1(y_{known},y_{true}) := 1/|y_{true}| \sum_{y\in y_{true}} \min_{x\in y_{known}} d(x,y), \tag{12}$$

$$dist_2(y_{known},y_{true}) := \max_{y\in y_{true}} \min_{x\in y_{knwon}} d(x,y), \tag{13}$$

wobei $d(x,y) := \max_{i\in\{1,2\}} \{(1/|q_i^{max} - q_i^{min}|)|q_i(x) - q_i(y)|\}$ gilt. Das Maß $dist_1$ bestimmt für jede Lösung $y \in y_{true}$ die am nächsten liegende Lösung in y_{known} und ermittelt dann den durchschnittlichen Abstand, während $dist_2$ für ein beliebiges $y \in y_{true}$ den kleinsten Abstand zu einer Lösung $x \in y_{known}$ sucht und dann das Maximum dieser Abstände verwendet.

Wir verwenden 350 Individuen in einer Population sowie 200 Generationen für NSGA-II bzw. NSGA-II-H. Die NSGA-II-artigen Verfahren wurden in C++ unter Verwendung des MOMHLib++-Rahmenwerks implementiert. LP-Solve wurde als MIP-Solver verwendet. Für die Mutation wird $p_{mut} = 0.4$ und $p_g = 0.1$ gewählt. Für die Initialisierung von NSGA-II-H wurde *g=100* verwendet. Zur Festlegung dieser Parameter wurden Pilotversuche durchgeführt. Fünf unabhängige GA-Wiederholungen wurden für jede Probleminstanz durchgeführt.

4.2 Ergebnisse für kleine Probleminstanzen

Für kleine Probleminstanzen ist es möglich, die wahre Pareto-Front y_{true} unter Verwendung der ε -Methode [3] zu bestimmen. Dazu wird das Modell (3)-(6) verwendet, wobei die Zielfunktion (3) durch $\sum_{i=1}^{n} q_1(s_{ij})\, z_{ij}$ ersetzt und die zusätzliche Nebenbedingung

$$\sum_{i=1}^{n} q_2(s_{ij})\, z_{ij} \leq \varepsilon_2^{(k)} \tag{14}$$

betrachtet wird. Das Modell wird iterativ gelöst. Die modifizierte Zielfunktion wird zur Minimierung des Wertes von q_1 verwendet. Durch Nebenbedingung (14) wird der maximale Wert des QoS-Attributs q_2 in Iteration k auf $\varepsilon_2^{(k)}$ eingeschränkt, wobei wir in der ersten Iteration $\varepsilon_2^{(1)} := \sum_{i=1}^{n} \max_{s_{ij} \in S_i} q_2(s_{ij})$ verwenden. Die optimale Lösung s_k des Modells in Iteration *k* minimiert den Wert von q_1 unter Beachtung von Nebenbedingung (5) und (14). s_k ist aber nicht notwendigerweise eine Pareto-optimale Lösung, da es eine zulässige Lösung s'_k geben kann mit $q_2(s'_k) \leq q_2(s_k)$ und $q_1(s'_k) = q_1(s_k)$. Zur Bestimmung von s'_k wird ein zweites Modell verwendet, das ebenfalls auf dem Modell (3)-(6) basiert, wobei die Zielfunktion (3) durch $\sum_{i=1}^{n} q_2(s_{ij})\, z_{ij}$ ersetzt und die folgende Nebenbedingung hinzugenommen wird:

$$\sum_{i=1}^{n} q_1(s_{ij})\, z_{ij} \leq \varepsilon_1^{(k)}. \tag{15}$$

Wir verwenden $\varepsilon_1^{(1)} := q_1(s_1)$. Die optimale Lösung des zweiten Modells s'_k ist ebenfalls eine zulässige Lösung für die betrachtete Probleminstanz. Da s'_k den Wert von q_2 minimiert, ist s'_k auch eine Pareto-optimale Lösung für diese Probleminstanz. In Iteration $k+1$ wird nun $\varepsilon_2^{(k+1)} := q_2(s'_k) - 1$ und $\varepsilon_1^{(k+1)} := q_1(s_{k+1})$ gesetzt. Wir weisen darauf hin, dass für unsere Probleminstanzen q_2 ganzzahlig ist. Das Verfahren wird beendet, sobald keine zulässige Lösung durch das erste Modell gefunden werden kann. Es wird zur Abkürzung mit E-M bezeichnet.

Das Verfahren wurde auf vier Probleminstanzen mit $n = 10$, $|S_i| = 10$, $C = 0.05$ und $c = -4$ angewandt. Die entsprechenden Ergebnisse sind in Tabelle 2 dargestellt.

	NSGA-II	NSGA-II-H	IP	E-M
Error	0,283	0,000	0,000	0,000
$dist_1$	0,028	0,000	0,000	0,000
$dist_2$	0,093	0,000	0,000	0,000
ONVG	8,750	12,500	12,500	12,500

Tabelle 2: Ergebnisse für kleine Probleminstanzen

NSGA-II-H findet für alle vier Probleminstanzen die wahre Pareto-Front. y_{IP} stimmt dabei jeweils mit $y_{NSGA\text{-}II\text{-}H}$ überein, da bereits in $y_{NSGA\text{-}II\text{-}H}$ alle Lösungen Pareto-optimal sind. $y_{NSGA-II}$ führt zu einem über alle vier Probleminstanzen gemittelten Fehler von 28%. Die Werte für $dist_1$ und $dist_2$ zeigen jedoch, dass die dominierten Lösungen nahe an den Pareto-optimalen Lösungen

liegen. In $y_{NSGA-II}$ sind weniger Lösungen als in y_{E-M} enthalten. Die Rechenzeit pro Instanz beträgt auf einem Pentium 4, 3.6 GHz PC mit 2 GB RAM für E-M bis zu 57 Sekunden, während für die übrigen Heuristiken 3 - 15 Sekunden erforderlich sind.

4.3 Ergebnisse für große Probleminstanzen

Für die Faktorkombinationen aus Tabelle 1 wurden jeweils zehn unabhängige Probleminstanzen erzeugt, insgesamt also 360 Instanzen. Als Referenzmenge y_{true} wird die Vereinigung der Pareto-Fronten $y_{true} := y_{NSGA-II} \cup_{nondom} y_{NSGA-II-H} \cup_{nondom} y_{IP}$ für jede Probleminstanz verwendet, wobei dominierten Lösungen entfernt werden. In Tabelle 3 sind die Mittelwerte der Werte der Leistungsmaße über alle Probleminstanzen dargestellt.

	NSGA-II	NSGA-II-H	IP
Error	0,971	0,709	0,000
$dist_1$	0,458	0,005	<0,001
$dist_2$	0,845	0,057	0,039
ONVG	2,700	122	101
Rechenzeit/Instanz (s)	13	14	860

Tabelle 3: Ergebnisse für große Probleminstanzen

Tabelle 3 zeigt deutlich, dass NSGA-II-H NSGA-II überlegen ist. Der $ONVG$ -Wert belegt, dass NSGA-II in vielen Fällen nicht in der Lage ist, viele zulässige Lösungen zu finden. Die hohen Werte für die restlichen Leistungsmaße zeigen, dass $y_{NSGA-II}$ deutlich weiter von y_{true} entfernt ist als $y_{NSGA-II-H}$. Die Werte für $dist_1$ und $dist_2$ sind größer als 0. Dieses Verhalten wird dadurch verursacht, dass durch NSGA-II bezüglich y_{true} Pareto-optimale Lösungen ermittelt werden, die von keiner IP-Berechnung, von Lösungen in NSGA-II-H ausgehend, gefunden werden. Der direkte Vergleich zwischen NSGA-II-H und IP im Hinblick auf $Error$ zeigt, dass für kleine Instanzen NSGA-II-H nahezu alle Pareto-optimalen Lösungen findet. Für größere Instanzen steigt der Wert von $Error$ auf bis zu 70%, wobei auch hier $dist_1$ und $dist_2$ zeigen, dass sich die Lösungen, die nicht Pareto-optimal sind, in der Nähe von Pareto-optimalen Lösungen befinden. Aufgrund der Konstruktion von IP ist offensichtlich, dass von IP weniger Lösungen gefunden werden als von NSGA-II.

5 Zusammenfassung und Ausblick auf weitere Arbeiten

Verschiedene Heuristiken zur Erweiterung von NSGA-II für $SCP(2,1,PF)$ wurden vorgeschlagen. Ein IP-basiertes Verbesserungsverfahren wurde untersucht, das zur Bestimmung von Pareto-optimalen Lösungen verwendet wird. Die problemspezifischen Erweiterungen von NSGA-II finden Lösungen hoher Qualität. Trotz einer deutlich besseren Abtastung der Pareto-Front durch NSGA-II-H werden bei großen Probleminstanzen viele dominierte Lösungen gefunden, die sich allerdings in der Nähe Pareto-optimaler Lösungen befinden. Falls Pareto-Optimalität erforderlich ist, muss folglich auf das IP-Verfahren zurückgegriffen werden.

Im Rahmen von weiteren Forschungsarbeiten sollen die vorgeschlagenen Lösungsverfahren für $SCP(m,r,PF)$ mit $m > 2$ und $r \geq 2$ verallgemeinert werden. Weiterhin werden in zukünftigen Forschungsarbeiten im Vergleich zu einer sequentiellen Dienstkomposition auch Kompositionen untersucht, in denen z.B. die parallele Ausführung von Tasks erlaubt ist.

6 Literatur

[1] Deb, K; Pratap, A.; Agarwal, S; Meyarivan, T. (2002): A fast and elitist genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2): 182-197.

[2] Deb, K; Goel, T (2001): A hybrid multi-objective evolutionary approach to engineering shape design. In: Zitzler, E; Thiele, L; Deb, K; Coello, CA; Corne, D (Hrsg.), Evolutionary Multi-Criterion Optimization. Zürich, Schweiz.

[3] Ehrgott, M (2010): Multicriteria Optimization. 2. Auflage. Springer, Berlin, Heidelberg, New York.

[4] Huhns, MN (2008): Services must become more agent-like. Wirtschaftsinformatik 50(1): 5-7.

[5] Jaeger, M; Muehl, G (2007): QoS-based selection of services: The implementation of a genetic algorithm. In: Braun, T; Carle, G; Stiller, B (Hrsg.), KiVS 2007 Workshop: Service-Oriented Architectures und Service Oriented Computing (SOA/SOC). Bern, Schweiz.

[6] Kellerer, H; Pferschy, U; Pisinger, D (2004): Knapsack Problems. Springer, Berlin, Heidelberg, New York.

[7] Li, W; Yan-Xiang, H (2010): A Web service composition algorithm based on global QoS optimizing with MOCACO. In: Hsu, CH; Yang, LT; Park, JH; Yeo, SS (Hrsg), Algorithms and Architectures for Parallel Processing. Busan, Korea.

[8] Liu, S; Liu, Y; Ning, J; Guifen, T; Yu, T (2005): A dynamic Web service selection strategy with QoS global optimization based on multi-objective genetic algorithm. In: Zhuge, H; Fox, GF (Hrsg.), Grid and Cooperative Computing - GCC 2005, 4th International Conference. Beijing, China.

[9] Papazoglou, MP; Traverso, P; Dustar, S.; Leymann, F. (2008): Service-oriented computing: a research roadmap. International Journal of Cooperative Information Systems 17(2): 223-255.

[10] Siedersleben, J (2007): SOA revisited: Komponentenorientierung bei Systemlandschaften. Wirtschaftsinformatik 49: 110-117.

[11] Veldhuizen, VA (1999): Multiobjective evolutionary algorithms: classification, analyses, and new innovations. PhD Thesis, Department of Electrical and Computer Engineering, Air Force Institute of Technology.

[12] Wada, H; Suzuki, J; Yamano, Y; Katsuya, O (2011): E3: Multi-objective genetic algorithms for SLA-aware service deployment optimization problem. IEEE Transactions on Services Computing, angenommen zur Veröffentlichung.

[13] Yu, T; Lin, K (2004): Service selection algorithms for Web services with end-to-end QoS constraints. Information Systems and E-Business Management 3(2): 106-126.

[14] Yu, T; Zhang, Y; Lin, K (2007): Efficient algorithms for Web services selection with end-to-end QoS constraints. ACM Transactions on the Web 1(1): paper 6.

[15] Zeng, L; Benatallah, B; Dumas, M; Kalagnanam, J; Sheng, Q (2003): Quality driven Web services composition. In: Hencsey, G; White, B (Hrsg), Proceedings of the 12th International Conference on World Wide Web. Budapest, Ungarn.

[16] Zitzler, E; Thiele, L (1998): Multiobjective optimization using evolutionary algorithms - a comparative case study. In: Eiben, A; Bäck, T; Schoenauer, M; Schwefel, H (Hrsg), Proceedings of the 5th International Conference on Parallel Problem Solving from Nature. Amsterdam, Niederlande.